REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 14-01-2015	2. REPORT TYPE Final Report	3. DATES COVERED (From - To) 15-Jun-2010 - 14-Jun-2014

4. TITLE AND SUBTITLE
Final Report: Modeling motivational and action attitudes

5a. CONTRACT NUMBER
W911NF-10-1-0250

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611102

6. AUTHORS
Yoav Shoham

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
Stanford University
3160 Porter Drive, Suite 100

Palo Alto, CA 94304 -8445

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
57902-NS.4

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
Time management – for both individuals and groups – is notoriously hard. The planning process is tedious, and the outcome is often inefficient for both the individual and the group. We seek to improve both the process and the outcome by transforming the digital calendar from a passive repository of events to an active scheduling assistant in four key steps. First, building on logical theories of intention, we enrich the expressive power of calendar entries to better capture the way people naturally think about intentions and plans. Second, building on optimization techniques and empirical findings from behavioral psychology, we develop scheduling algorithms to efficiently

15. SUBJECT TERMS
intelligent scheduling, logic, optimization, human factors, algorithms, software systems, digital calendars

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Yoav Shoham
UU	UU	UU	UU		19b. TELEPHONE NUMBER 650-723-3432

Standard Form 298 (Rev 8/98)
Prescribed by ANSI Std. Z39.18

Report Title

Final Report: Modeling motivational and action attitudes

ABSTRACT

Time management – for both individuals and groups – is notoriously hard. The planning process is tedious, and the outcome is often inefficient for both the individual and the group. We seek to improve both the process and the outcome by transforming the digital calendar from a passive repository of events to an active scheduling assistant in four key steps. First, building on logical theories of intention, we enrich the expressive power of calendar entries to better capture the way people naturally think about intentions and plans. Second, building on optimization techniques and empirical findings from behavioral psychology, we develop scheduling algorithms to efficiently allocate time. Third, we extend our new calendar entries and scheduling algorithms to the multiagent setting. Finally, we are in the process of developing software that combines this theoretical work with good interface design and engineering to make a truly useful system.

In the reported funding period we concentrated on three tasks:
- Experimental study of human productivity patterns. In partnership with Rescue Time (www.rescuetime.com) we have collected productivity patterns of several hundred users, and plan to analyze it using machine learning to discern productivity type that go beyond simplistic categories such as "morning person / evening person".
- Algorithmic study of optimal group scheduling. Group scheduling is a notoriously painful process, both in terms of the process and the quality of the outcome. We initiated an algorithmic study of optimal group scheduling, with an eye towards improving on common tools such as Doodle (www.doodle.com). We have already obtained some results, both positive and negative.
- Transitioning of the intelligent calendar software to a commercial spinoff. We applied for a patent in connection with our early work on the flexible calendar, and a company -- headed by one of the PhD students -- was established to commercialize the ideas.

Enter List of papers submitted or published that acknowledge ARO support from the start of the project to the date of this printing. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in peer-reviewed journals:

(b) Papers published in non-peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in non peer-reviewed journals:

(c) Presentations

Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received		Paper

TOTAL:

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received		Paper
01/14/2015	2.00	Yoav Shoham, Hooyeon Lee. Stable Invitations, AAAI 15. 26-JAN-15, . : ,
01/14/2015	3.00	Hooyeon Lee, Yoav Shoham. Optimizing time and convenience in group scheduling, AAMAS 14. 07-MAY-14, . : ,
08/02/2012	1.00	Jacob Bank, Zachary Cain, Yoav Shoham, Caroline Suen, Dan Ariely. Turning Personal Calendars into Scheduling Assistants, CHI 2012. 07-MAY-12, . : ,

TOTAL: **3**

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts):

(d) Manuscripts

Received	Paper

TOTAL:

Number of Manuscripts:

Books

Received	Book

TOTAL:

Received	Book Chapter

TOTAL:

Patents Submitted

Patents Awarded

Awards

Graduate Students

NAME	PERCENT_SUPPORTED	Discipline
Jacob Bank	0.04	
Hooyeon Lee	0.04	
Alice Chung	0.13	
Zack Weiner	0.10	
FTE Equivalent:	**0.31**	
Total Number:	**4**	

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
Yoav Shoham	0.16	Yes
FTE Equivalent:	**0.16**	
Total Number:	**1**	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED	Discipline
Tyler Strand	0.06	Computer and Computational Sciences
Melissa Johnson	0.05	Computer and Computational Sciences
FTE Equivalent:	**0.11**	
Total Number:	**2**	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period:	2.00
The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:......	2.00
The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:......	0.00
Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):......	0.00
Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense	0.00
The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields:......	0.00

Names of Personnel receiving masters degrees

NAME
Total Number:

Names of personnel receiving PHDs

NAME	
Total Number:	

Names of other research staff

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Sub Contractors (DD882)

Inventions (DD882)

Scientific Progress

See attachment.

Technology Transfer

Modeling Motivational and Action Attitudes

Progress Report: August 2011 - July 2012

Abstract

Time management - for both individuals and groups - is notoriously hard. The planning process is tedious, and the outcome is often inefficient for both the individual and the group. We seek to improve both the process and the outcome by transforming the digital calendar from a passive repository to an active scheduling assistant in four key steps. First, building on logical theories of intention, we enrich the expressive power of calendar entries to better capture the way people naturally think about their plans. Second, building on optimization techniques and findings from behavioral psychology, we develop scheduling algorithms to efficiently allocate time. Third, we extend our new calendar entries and scheduling algorithms to the multiagent setting. Finally, we are in the process of developing software that combines this theoretical work with good interface design and engineering to make a truly useful system.

Contents

1 Introduction

As everyone knows from personal experience, scheduling a meeting for even a small number of people is a frustrating process, typically involving complex negotiations and endless email chains. One would think that technology could ease the pain, and to some extent it does with tools such as shared calendars [15, 28] and Doodle [9], but this process of group coordination remains fundamentally manual and painful. And group scheduling is not the only pain-point in time management. Individuals spend significant time scheduling their own tasks and do it quite poorly: we procrastinate, we prioritize incorrectly, and we fall victim to a number of other poor scheduling habits [1, 25]. When we look at time management as a whole, the process is time-consuming and aggravating, and the outcomes are inefficient. Automating this process seems like an ideal solution, and indeed there have been many attempts to automate both meeting scheduling and individual task scheduling, but prior efforts (e.g. [4, 14, 23, 32, 34]) have not quite solved the problem [5].

Automated scheduling is riddled with challenges, the first of which is that such scheduling relies on digital calendars, and digital calendars are usually a poor representation of the user's intentions and plans [31]. Even if a calendar truly did represent a users full set of intentions, however, automated scheduling would be no easy task. First, a good schedule must take into account many subtle factors, including human psychology and productivity patterns, and the dynamic nature of plans. Second, coordination across multiple calendars needs to take additional, inter-personal considerations into account, which are also subtle. Because of the above difficulties people are extremely possessive of their time management and reluctant to cede control of their calendar to an automated (or even human) assistant, especially one that does not leave them (the users) in control of key decisions [6].

To solve the scheduling problem, we propose rethinking the digital calendar, transforming it from a passive repository of events into an active, intelligent scheduling assistant. Our approach is based on three principles:

1. Current basic calendar entries, which are essentially a description and a time slot, have the advantage of simplicity, but they are too crude to capture the many nuances of human intention. One needs more expressive calendar entries.
2. To translate this richer set of calendar entries into concrete events, a calendar needs highly nuanced algorithms that optimize for the productivity and enjoyment of individuals and groups.
3. The automation must be clear to the user, and the user must be left in control of key steps in the scheduling process.

We are developing a system, called Kairos (pronounced KYE-ross, an Ancient Greek word meaning "opportune time"), which is a web application that has the look and feel of a standard digital calendar but includes an extra layer of intelligence. In particular, it features a richer repertoire of calendar entry types and optimization algorithms for both individual and group scheduling. Although Kairos has not yet been widely deployed, we have conducted initial experiments and surveys. The responses, preliminary as they are, have been extremely encouraging.

The structure of this report is as follows. We first describe our richer set of calendar entries that improve the expressive power of the digital calendar. We then detail the methods for productivity-maximizing individual scheduling and group welfare-maximizing meeting scheduling. We follow that with a report of our initial user evaluations of the system. We then survey the related work on calendaring and automated scheduling. Finally, we discuss directions for future work.

2 Novel Calendar Entries

On the face of it, a calendar is a simple object – a record of one's future plans and past activities – but the reality is much more complex. For one thing, traditional calendars often under-represent a user's commitments. These commitments fail to appear on the calendar either because they are too tentative or vague to be made concrete ("I probably will need some time to prepare for that meeting"), because they are too personal to coldly specify ("spend time with kids", "take time to decompress"), or simply because the user never got around to updating the calendar. But calendars can also *over*-represent the person's constraints. When a user enters "grade homeworks 10am-12pm", it is unclear whether this choice was made definitively, or if the user would have been equally happy with another two-hour time slot.

Most digital calendars allow one to represent only what we might call *definitive events*: (what, when)-pairs, where "what" is a descriptor (such as "lunch") with some optional attributes (such as location or other participants), and "when" is a specific time slot ("10am-11am on Monday, May 28, 2012"). The definitive event has the advantage of simplicity and clarity, but most people use more complex cognitive structures when they think about the future; plans are often under-specified, tentative, and flexible. We focus specifically on the temporal flexibility of plans, taking a cue from the formal literature on intention [8, 16, 35]. So-called *atomic intentions* correspond to the standard calendar's definitive events: a particular activity at a particular time. But some intentions underspecify time ("do laundry for an hour sometime in the afternoon"), and some – corresponding to items from typical to-do lists – are even more open ended ("Complete a paper by next Friday"). These categories of events motivate Kairos to embrace temporal open-endedness by supplementing definitive events with new calendar entries, creating the following three event types[1].

1. *Simple Events* are the familiar calendar entries that have a manually specified time and duration ("I have a presentation on Monday from 1pm to 2pm").
2. *Floating Events* are a generalization of simple events to allow for temporal flexibility. These events have a known duration, but can occur at any time in a set of specified time windows ("I need to do laundry on Monday or Tuesday evening, and it takes 2 hours").[2]
3. *Tasks* are a structured extension of to-do items that capture their goal-oriented nature in a more actionable way. These are intentions with a known or unknown duration that may or may not have a deadline ("I have a paper due on June 1st that will take 15 hours to complete").

With this more expressive set of calendar entries, a user can now more easily specify their full set of intentions without losing the temporally flexible nature of these plans. By giving the digital calendar a more complete representation of a user's plans and intentions, we can both transfer cognitive load from the human to the software tool, but also increase the power of the software to act as a personal assistant.

[1]In an earlier version of Kairos we experimented with additional event types, including "conditional events and "multiple-choice events, but since in user evaluations those did not prove useful, they are not part of Kairos and we don't discuss them further here.

[2]Simple events are a special case of floating events, but we keep them separate because they are both familiar from previous calendar systems and extremely common.

3 Individual Scheduling

The addition of Kairos' new entry types creates both a challenge and an opportunity. The challenge is that the flexble entries now represent a huge space of possible instantiations and it is difficult for a human user to commit to a concrete schedule. The opportunity, however, is that we can make use of algorithms to help people search through this space, and in the process we can avoid some of the systematic mistakes people are prone to make in time management.

To create an automated scheduling agent, we draw on extensive productivity research, which has given us a set of principles an algorithm should adhere to in creating human-optimized schedules.

- Important tasks should be prioritized [25].
- Human performance rises and falls throughout the day, following circadian rhythms, and a schedule should try to match task difficulty with cognitive ability [7].
- Humans perform better on long tasks when they are split into manageable segments, as long as the task segments are of sufficient length to ramp up into the task [1].
- On creative tasks, incubation time between segments improves performance [27].
- Positive affect improves creativity [17].
- Travel time should be minimized.

We incorportate the above considerations into an optimization problem as follows. Given a set of N entities $E = e_1, e_2, ..., e_n$ and a matrix of temporal distances between all pairs of locations, output a schedule S of entities in which each entity e_i is described by p_i, the number of parts, t_{ij}, the start time of the jth part, and d_{ij}, the duration of the jth part to:

maximize $\sum_{e \in E} U(e) - D(e)$

subject to:

C1. $\forall i \forall j \ \sum_{j=1}^{p_i} d_{ij} = duration(e_i)$
(The sum of durations of segments will sum to the entity's duration, meaning the task is completed.)

C2. $\forall i \forall j \ minpart(e_i) <= d_{ij} <= maxpart(e_i)$
(Each segment's duration is between the minimum and maximum allowed duration for that entity.)

C3. $\forall i \forall j \forall k \ j < k \rightarrow t_{ij} + d_{ij} + incubation(e_i) \leq t_{ik}$
(Enough incubation time is left between segments of the same task.)

C4. $\forall i \forall j \ start(e_i) \leq t_{ij} \leq deadline(e_i) - d_{ij}$
(All segments are scheduled after the start time and before the deadline of the entity.)

C5. $\forall i \forall j \forall m \forall n \ t_{ij} < t_{mn} \Rightarrow t_{ij} + d_{ij} + Dist(e_i, e_n) \leq t_{mn}$
(Ensures that it is possible to travel between locations of temporally adjacent events in the allotted time.)

Where the functions have the following meaning:

- $U(e)$ is the utility function that outputs a productivity score for the scheduling of an entity.

$U(e)$ depends on the type of entity, as shown in the following function definition.

$$U(e) = \begin{cases} \frac{R}{travel(e)+1} & \text{for floating} \\ \\ \frac{s(e)*t(e)*p(e)}{travel(e)+1} & \text{for tasks} \end{cases}$$

R is a large real that ensures that floating events will always receive higher utility than task segments, since task segments are more flexible.

- $D(e)$ penalizes the magnitude of change from the previous schedule, as large changes can be jarring to users.
- $travel(e)$ is the total travel time required, and since less travel time is better, this term appears in the denominator.
- $s(e)$ describes how satisfying the scheduling of this task is.
- $t(e)$ describes how well the task scheduling matches cognitive ability throughout the day.
- $p(e)$ gives high scores to important tasks and low scores to unimportant tasks.
- $duration(e)$ is the duration of the task or event.
- $minpart(e)$ and $maxpart(e)$ control segment duration.
- $incubation(e)$ is the time required between segments.
- $start(e)$ and $deadline(e)$ are the start time and deadline.
- $Dist(e_1, e_2)$ is the travel time between e_1 and e_2.

With this formulation, we achieve all of the necessary features to make a schedule tuned to human performance, and in the future we imagine we will get large gains by personalizing to the individual. Since the search space for this optimization problem is exponential in size, we use heuristic methods that have been shown to approximately solve optimization problems like this one efficiently. First, since it is unnecessary and expensive to schedule all future entities, the algorithm only considers the entities relevant to the upcoming two week period (as determined by a simple heuristic of remaining time and deadline). Next, we follow prior work [33] and use a modified version of the Squeaky Wheel Optimization (SWO) [18] framework to solve the scheduling optimization problem.

A key issue in the creation of automated scheduling tools for the individual is that people are extremely possessive of their time and loathe to relinquish control of it to others. Any algorithmic scheduler must therefore make it very clear to the user what its doing and why, and allow the user to modify the scheduling according to personal preference. The automated schedule should act more as a first draft to guide the user in effective time management than as a rigid mandate of what the user must do. After the scheduling algorithm has terminated the user has the ability to move, extend, or delete any of the scheduled instances to fit their personal preferences, making the algorithm more of an intelligent rough draft than a binding mandate.

4 Multiagent Scheduling

One of the key goals of Kairos is to improve the tedious process of scheduling meetings by adding intelligent automation at the most painful steps. To do this, we follow the same approach as taken above in individual scheduling and formulate meeting scheduling as an optimization problem. The meeting scheduling problem can be expressed as an unconstrained optimization problem as follows. Given a desired duration d, a set of possible time windows W, and a set of attendees A, select a start time t to maximize the utility of the meeting. More formally:

Let $m = \{d, W, A\}$ be the meeting specification. maximize $U(t, m)$ where

$$U(t,m) = w_1 \left(\frac{\sum_{a \in A} importance(a) * availability(a,t)}{\sum_{a \in A} importance(a)} \right) + w_2(\min_{a \in A} idleness(a,t)) + w_3(\min_{a \in A} preference(a,t))$$

For clarity, we now break down each of the three components of the utility function. The first term captures the availability and importance of the attendees. $importance(a)$ is a function that returns the importance of the attendee to the meeting, on a scale from 1-5, and $availability(a, t)$ is a function that returns 0 if attendee a is unavailable and 1 if attendee a is available. Kairos automatically scans each user's calendar to extract availability. For each attendee we multiply the importance by whether the attendee can attend, sum those values, then divide by the sum of importances, which is the best value we could have achieved if every attendee was available. In this way we try to determine what proportion of the "importance mass" is available at this meeting time. This term gives us a value between 0 and 1, which we then weight by a constant w_1.

The next term captures the idleness of the attendees in the time surrounding the meeting, and rewards meeting times around which atiettendees are very free. $idleness(a, t)$ is a function that ranges from 0 to 1 and captures how idle attendee a is around time t. This term then adds the minimum attendee idleness to the utility function, meaning that we get small gains if the busiest attendee is very busy, and large gains if the busiest attendee is very free. We then weight this term by a constant, w_2.

The final term captures the preferences of the attendees for the given time slot. Since different people prefer to meet at different times of the day. $preference(a, t)$ is a function that ranges from 0 to 1 and captures attendee a's degree of preference for time t. The term adds the minimum of this function over all attendees to the utility function, meaning that we get gains proportional to how happy the least happy attendee is with the meeting time. We also weight this term by a constant, w_3.

Since availability and importance are most crucial, we set w_1 to .7. We hypothesize that preferences are next most important, and assign w_3 to .2, and finally w_2 for idleness is .1. In the futkairosure we plan to learn these weights from user behavior.

Solving this optimization problem is actually quite easy; we simply iterate over each possible time in each window, compute the utility function for this time (which outputs a quality score between 0 and 1), then sort the results by utility and output the top slots for the user to make a final selection.

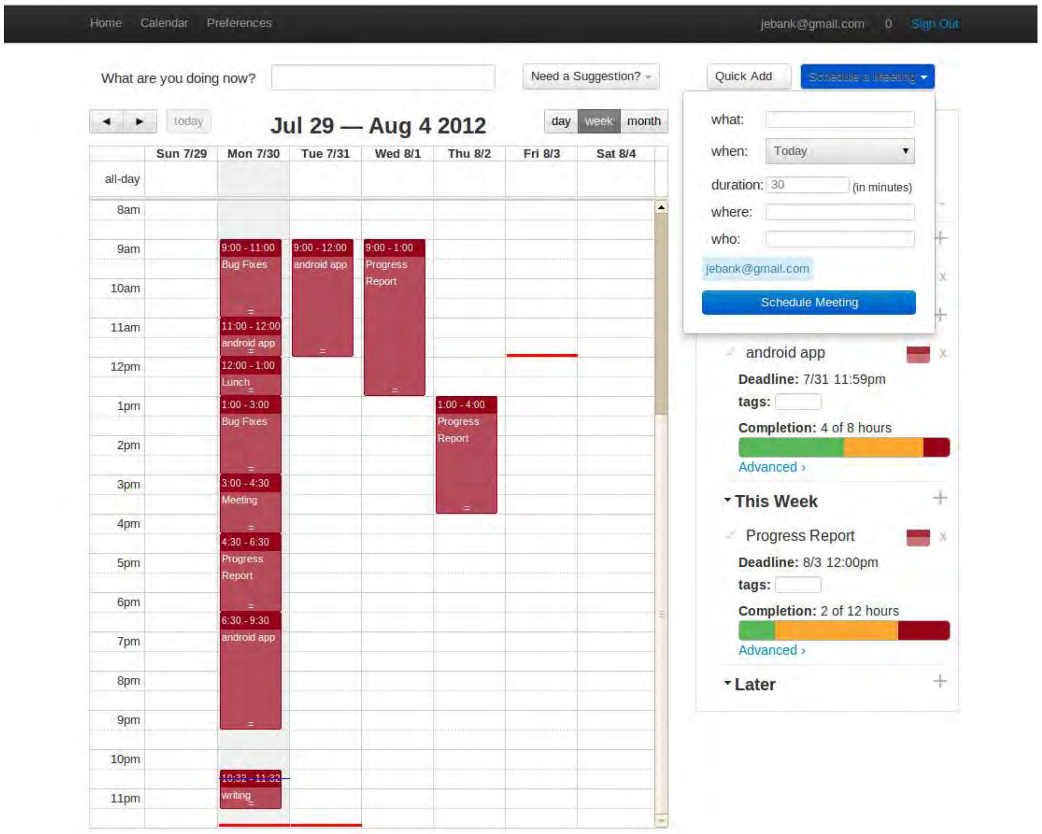

Figure 1: The main page of the Kairos system.

5 Software System

Based on these principles, we are in the process of implementing a web application, called Kairos. The system functions as an integrated digital calendar and todo list, with our new algorithms and event specifications as an additional layer on top of standard functionality.

As shown in Figure 1, Kairos has the look and feel of a standard digital calendar with a task list on the right. To add events, one simply clicks on the calendar, drags the length of the event, then inputs a title. To add tasks, the user can either use the add a task bar, or click the plus sign next to the time buckets ("Today", "Tomorrow", etc). To schedule a task using the scheduling algorithm, the user opens the advanced view of the task in the task bar and clicks the schedule button. The remaining duration of that task is then scheduled.

To initiate a meeting, the user clicks the "Schedule a Meeting" button in the top right, which then drops down a meeting specification box as seen in Figure 1. The initiator of the meeting specifies the details of the meeting, including the time boundaries and the attendees, then initiates the scheduling algorithm. After the scheduling user chooses one of the top slots, the attendees are all notified and given the options to confirm attendance, request that the meeting be rescheduled, or decline to attend the meeting.

The latest version of the software, which is a Python web application running on Google Appengine, can be found at http://kairos-calendar.appspot.com.

6 Preliminary Evaluation

To gauge user response to the Kairos project, we used two disjoint surveys (with different subjects), one on personal scheduling, and the other on meeting scheduling. To test the personal scheduling features, we had 15 college students test Kairos in a exercise designed to simulate time management in a typical week. For the exercise, we populated the calendar with a few mandatory events and then provided a long list of possible activities. Some of these were work obligations, others were social, and they had varying properties of importance, difficulty, and duration. After a brief description of Kairos' functionality, the students had 20 minutes to live through this hypothetical week at an accelerated pace, scheduling tasks and events on the calendar, with the help of the scheduling algorithm as desired. After the subjects completed the exercise, we asked them to qualitatively respond to Kairos' features through a questionaire. Tables 1-6 show the questions and responses.

Task	Simple	Floating
73%	20%	7%

Table 1: Answers to: "Which entity type is most useful?"

Task	Simple	Floating
39%	40%	21%

Table 2: Answers to: "If you were to enter all of your intentions into Kairos, what percentage of each entity type do you think you would use?".

Perfectly	Well	Adequately	Poorly
27%	60%	13%	0%

Table 3: Answers to: "How well did the digital calendar allow you to express your intentions for the week?".

Worse	About the Same	Better
0%	60%	40%

Table 4: Answers to: "How do the schedules produced by Kairos compare to the schedules you typically produce?"

Yes	No
100%	0%

Table 5: Answers to: "Would you be willing to allow a scheduling algorithm to help you manage your time?".

Yes	No
93%	7%

Table 6: Answers to: "Do you think a scheduling algorithm to help manage time would make you more productive".

To test the meeting scheduling features, we distributed another survey. 61 respondents began the survey, and 43 completed it. 17 of the respondents are current college students (graduate and undergraduate), 10 are assistants who manage the calendars of busy executives, and the remaining 16 are tech workers that manage their own calendars. The ages of the calendar users ranged from 19 to 55. In the survey, we first asked a set of questions to assess basic calendar usage, then we showed a five minute demo video of Kairos meeting scheduling features, followed by questions about these features. Tables 7-9 show the responses to questions about the individual features of the meeting scheduling system, and most importantly, Table 10 shows the overall reactions.

Worse	About the Same	Better
5%	21%	74%

Table 7: Answers to:"Compared to the way you currently schedule meetings, is Kairos' automatic determination of participant availability worse, better, or about the same?"

Worse	About the Same	Better
5%	17%	78%

Table 8: Answers to:"Compared to the way you currently schedule meetings, is Kairos' system of participants' preference inclusion worse, better, or about the same?"

Worse	About the Same	Better
2%	12%	86%

Table 9: Answers to:"Compared to the way you currently schedule meetings, is Kairos' method of presenting a ranked list of possible meeting slots with rationale for that ranking worse, better, or about the same?"

Much Worse	Worse	Same	Better	Much Better
0%	7%	17%	55%	21%

Table 10: Answers to:"Overall, how would you compare Kairos' meeting scheduling to the way you currently schedule meetings?"

While these evaluations are quite preliminary in that the surveys were not performed on real users of the system, but rather users that had seen a demonstration, the positive responseses demonstrate the promise of our approach. Potential users seem to appreciate the gain in expressive power provided by the new calendar entries, and they also seem to be open to receive scheduling help from an automated agent.

7 Related Work

Calendaring and scheduling have been studied extensively, and we build on this work both in the creation of our new calendar entries and in our scheduling algorithms. In addition to covering the works that directly inspired features of Kairos, this section also provides a general survey of work on calendaring.

To improve the expressive power of our calendar, we build on theoretical work on intention and previous studies of calendar usage to derive our new calendar entries. The theory of intention, anchored in the work of Cohen and Levesque [8] and extended by Shoham [35] and Icard et al. [16], gave us the basic framework of atomic intentions and temporal flexibility. The idea to incorporate to-do and flexible items further grows from previous studies of calendar and task manager use, including a study on task management and design considerations from Bellotti et al [3], a study of the inadequcy of time management tools by Blandford and Green [5], and Payne's observations on the inability of calendars to represent conceptual intentions [31].

To create our scheduling algorithms, we combine work from experimental psychology with prior work on optimization algorithms for scheduling. Human productivity under different conditions has been studied for many years, and we build on a few branches of this work to establish the properties of a productive schedule. As procrastination is one of the most damaging errors in time management, we first reference recent work that has documented the negative effects of procrastination on performance. In one study, O'Donoghue and Rabin [25] showed that people procrastinate more on important tasks, which inspired us to prioritize important tasks in scheduling. Motivation for breaking tasks into segments came from work by Ariely and Wertenbroch [1], which demonstrated that a lack of intermediate deadlines decreases performance. In addition to experiments on procrastination, we also drew heavily on work on creative problem solving. In a study on the spacing of work on such tasks, Olton and Johnson [27] showed that incubation time between sessions of work boosts performance. In a study of the effect of emotional state on creative ability, Isen et al [17] showed that a positive state of mind improves results on tasks requiring originality. Finally, we draw on a study from biopsychology in which Carrier and Monk [7] documented how cognitive ability rises and falls throughout the day.

Similarly, scheduling and optimization have been studied in many contexts. The general framework and algorithm we use to solve the scheduling problem is Squeaky Wheel Optimization (SWO) from Joslin and Clements [18]. In using SWO, we follow previous calendar work by Refanidis, Yorke-Smith, and Alexiadis [33, 32], which presented scheduling in digital calendars as an optimization problem. In their work, they demonstrated that their slightly modified SWO algorithm solves the scheduling optimization problem effectively and efficiently. Our optimization formulation and SWO solving algorithm are directly inspired by their work [33]. While they deserve credit for their pioneering work, our framework is unique in its particular calendar entity types and the incorporation of psychological considerations

In addition to the specific work that motivates our approach, many other dimensions of calendaring have been previously explored. In early work on the topic, Kelley and Chapanis [19] interviewed professionals to understand the uses of personal calendars and ways in which the functionality could be digitized, finding, among other things, that a substantial number of appointments are changed after initial scheduling (showing the value of our flexible entities). In [29], Palen discussed how the analysis of groupware calendar systems requires understanding both the individual's use of the technology and the social context in which that use occurs to be able to extract the most from digital time management assistants. Then Palen and Grudin [30] examined adoption of groupware in enterprise settings and the challenges of effectively deploying shared calendaring systems.

In more recent qualitative work, Neustaedter et al [24] explored the role of personal calendars in the family, doing a detailed study of how people use personal calendars in all contexts of life and how families coordinate events and tasks using calendars. In [36], Tungare et al did an exploratory study of personal calendar use, discussing the nature of calendar events and also why people continue to use paper calendars. In another study on time management, Leshed and Sengers [21] analyzed how our culture of busyness comes from more than the desire to just get things done, and that productivity tools include social and emotional dimensions. In a study on time management in the enterprise [10], Dugan et al created simple visualizations of how people spend their time in the workplace to give them actionable insights about their time.

In addition to empirical observations on calendar use, a number of research systems have been developed to serve as automated personal assistants and meeting schedulers. Modi et al [23] developed a personal assistant called CMRadar designed to do calendar management, but that system primarily focused on interpreting emails and negotiating for meeting scheduling. Another major project in this space is the Cognitive Assistant that Learns and Organizes (CALO) [4], which focuses on creating a proactive agent to assist in task management and delegation, but does not include flexible entity types, nor does it include psychology in task scheduling. In [6], the authors design a lightweight interaction model for expression preferences of meeting times, to then be used for scheduling meetings via email. Geyer et al [14] developed a lightweight scheduling system in the model of social media, with subscriptions, social circles, and broadcasting of scheduling desires. Wang et al [38] develop a collaborative system for large events, which focuses on organizing large, multitrack conferences, including collaborate aspects.

In addition to these more recent systems, multiagent meeting scheduling has been studied in great depth by the AI community, especially from the perspective of creating individual negotiation agents and distributed optimization. In early work [13, 34], the authors conducted experiments on meeting scheduling agents, analyzing negotiation strategies and convergence patterns. Other work [11, 12, 37] studied the effects of maintaining privacy on scheduling efficiency and solution quality. Methods for learning user preferences [20, 22, 26] have also been studied extensively.

8 Future Work

The results of our initial surveys demonstrate that this approach to an active calendar is promising, but encouraging as these results are, ultimate validation will only come when many users begin to use our system outside the lab to represent and schedule their daily activities. As the development of our system continues, we plan to initiate longer term studies geared at understanding more about how people manage time and represent their intentions. As we get user feedback, we also plan to adapt our calendar entry types and possibly add new ones.

Another key strand of our future work will be to personalize the scheduling algorithms. We know that people prefer to schedule different types of tasks in different ways. Some people like longer segments to allow themselves to fully engage in a task, while others like to context-switch often to stay sharp. Some people like different amounts of spacing between segments of tasks, or different sequences of types of tasks to make them happiest and most productive. To achieve this personalization, we plan to develop a machine learning framework that will customize to a user's preferences over time using both implicit and explicit feedback.

In addition to efficiently finding a good slot for a meeting, Kairos should also aid in determining whether a meeting should occur at all. A constant complaint in the workplace is that people are trapped in too many meetings, leaving little time to accomplish actual work. Because Kairos can capture the full set of each user's tasks and events, the system can determine the collective opportunity cost of attending the meeting, allowing the meeting organizer to make a better judgement as to whether the meeting is the best use of time. We also intend to devise a system to poll the attendees of a meeting about its utility in hindsight, allowing us to construct a learning framework for meeting value. By combining opportunity cost and analytics of past meetings, we hypothesize that we can reduce the number of unnecessary meetings.

Once Kairos is deployed with real users, it will give us a tremendous opportunity to collect data on how people spend their time, and how time management can be improved. We plan to implement systems to poll subject feelings of productivity and happiness throughout the day, and use these measurements to identify the tradeoffs between productivity and happiness and to understand the types of activities and schedules that lead to high quality outcomes.

References

[1] D. Ariely and K. Wertenbroch. Procrastination, deadlines, and performance: Self-control by precommitment. *Psychological Science*, 13(3):219, 2002.

[2] Jacob Bank, Zachary Cain, Yoav Shoham, Caroline Suen, and Dan Ariely. Turning personal calendars into scheduling assistants. CHI EA, 2012.

[3] Victoria Bellotti, Brinda Dalal, Nathaniel Good, Peter Flynn, Daniel G. Bobrow, and Nicolas Ducheneaut. What a to-do: studies of task management towards the design of a personal task list manager. CHI, 2004.

[4] Pauline M. Berry, Melinda T. Gervasio, Bart Peintner, and Neil Yorke-Smith. Ptime: Personalized assistance for calendaring. *ACM TIST*, 2(4), 2011.

[5] A. E. Blandford and T. R. G. Green. Group and individual time management tools: What you get is not what you need. *Personal Ubiquitous Computing*, 5(4), 2001.

[6] Mike Brzozowski, Kendra Carattini, Scott R. Klemmer, Patrick Mihelich, Jiang Hu, and Andrew Y. Ng. grouptime: preference based group scheduling. CHI, 2006.

[7] J Carrier and T H Monk. Circadian rhythms of performance: new trends. *Chronobiology International*, 17(6), 2000.

[8] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3), 1990.

[9] Doodle. http://www.doodle.com/.

[10] Casey Dugan, Werner Geyer, Michael Muller, Abel N. Valente, Katherine James, Steve Levy, Li-Te Cheng, Elizabeth Daly, and Beth Brownholtz. "i'd never get out of this !?$%# office": redesigning time management for the enterprise. CHI, 2012.

[11] M. S. Franzin, E. C. Freuder, F. Rossi, and R. Wallace. Multi-agent meeting scheduling with preferences: Efficiency, privacy loss, and solution quality. AAAI Workshop on Preference in AI and CP, 2002.

[12] Eugene C. Freuder, Marius Minca, and Richard J. Wallace. Privacy/efficiency tradeoffs in distributed meeting scheduling by constraint-based agents. IJCAI, pages 63–72, 2001.

[13] Leonardo Garrido and Katia Sycara. Multi-agent meeting scheduling: Preliminary experimental results. ICMAS, 1996.

[14] Werner Geyer, Casey Dugan, Beth Brownholtz, Mikhil Masli, Elizabeth Daly, and David R. Millen. An open, social microcalender for the enterprise: timely? CHI, 2011.

[15] Google Calendar. http://www.calendar.google.com/.

[16] T. Icard, E. Pacuit, and Y. Shoham. Joint revision of belief and intention. KR, 2010.

[17] A. M. Isen, K.A. Daubman, and G. P. Nowicki. Positive affect facilitates creative problem solving. *Journal of Personality and Social Psychology*, 52(6), 1987.

[18] David Joslin and David P. Clements. Squeaky wheel optimization. *J. Artif. Intell. Res. (JAIR)*, 10, 1999.

[19] J. F. Kelley and A. Chapanis. How professional persons keep their calendars: Implications for computerization. *Journal of Occupational Psychology*, 55, 1982.

[20] Robyn Kozierok and Pattie Maes. A learning interface agent for scheduling meetings. IUI, 1993.

[21] Gilly Leshed and Phoebe Sengers. "i lie to myself that i have freedom in my own schedule": productivity tools and experiences of busyness. CHI, 2011.

[22] Pattie Maes and Robyn Kozierok. Learning interface agents. In *AAAI*, 1993.

[23] Pragnesh Jay Modi, Manuela Veloso, Stephen F. Smith, and Jean Oh. Cmradar: A personal assistant agent for calendar management. AOIS, 2004.

[24] Carman Neustaedter, A. J. Bernheim Brush, and Saul Greenberg. The calendar is crucial: Coordination and awareness through the family calendar. *ACM Trans. Comput.-Hum. Interact.*, 16(1), 2009.

[25] Ted O'Donoghue and Matthew Rabin. Choice and procrastination. *The Quarterly Journal of Economics*, 116(1), 2001.

[26] Jean Oh and Stephen F. Smith. Learning user preferences in distributed calendar scheduling. PATAT, 2004.

[27] Robert M. Olton and David M. Johnson. Mechanisms of incubation in creative problem solving. *The American Journal of Psychology*, 89(4).

[28] Microsoft Outlook. http://www.office.microsoft.com/en-us/outlook/.

[29] Leysia Palen. Social, individual and technological issues for groupware calendar systems. CHI, 1999.

[30] Leysia Palen and Jonathan Grudin. Implementing collaboration technologies in industry. chapter Discretionary adoption of group support software: lessons from calendar applications. 2003.

[31] Stephen J. Payne. Understanding calendar use. *Hum.-Comput. Interact.*, 8(2), 1993.

[32] Ioannis Refanidis and Anastasios Alexiadis. Deployment and evaluation of selfplanner, an automated individual task management system. *Computational Intelligence*, 27(1), 2011.

[33] Ioannis Refanidis and Neil Yorke-Smith. A constraint-based approach to scheduling an individual's activities. *ACM TIST*, 1(2), December 2010.

[34] Sandip Sen and Edmund H. Durfee. A formal study of distributed meeting scheduling. *Group Decision and Negotiation*, 7.

[35] Y. Shoham. Logical theories of intention and the database perspective. *Journal of Philosophical Logic*, 38(6), 2009.

[36] Manas Tungare, Manuel A. Pérez-Quiñones, and Alyssa Sams. An exploratory study of calendar use. *CoRR*, abs/0809.3447, 2008.

[37] Richard J. Wallace and Eugene C. Freuder. Constraint-based reasoning and privacy/efficiency tradeoffs in multi-agent problem solving. *Artificial Intelligence*, 161(1-2), 2005.

[38] Jingtao Wang, Danny Soroker, and Chandra Narayanaswami. Event maps: a collaborative calendaring system for navigating large-scale events. CHI EA, 2010.

Research Results

September 20, 2013

Our research in the period focused on two topics: experimental study of human productivity patterns, and algorithmic study of group scheduling.

The report on the first topic is brief. The goal is to study ways in which the intelligent calendar can make recommendation based on people's individual productivity cycles. There are general rules of thumb such as "don't schedule high concentration tasks right after lunch", and some coarse-grade individual personalities ("morning person" versus "evening person"). The question is whether a system could get a much more personalized sense of a person's daily rhythm, going far beyond these simplistic categories. To study this we initiated a study using Rescue Time (http://www.rescuetime.com). The rescue time application tracks a user's activities on the desktop, and tracks his/her usage of productive applications (such as software tools) versus nonproductive applications (Facebook, say). By agreement with the company, we recruited several hundred users to use the application and have their usage data sent to us, where we could correlate it with demographic and other data they supplied us and, using machine learning techniques, infer some interesting user productivity types. We recently completed the data collection phase, and plan to commence the analysis phase shortly.

Research on the second topic is farther along; we have some concrete results, some of which were already presented at EC'13 (ACM Conference on Electronic Commerce). We discuss the results in more detail in the following sections.

1 Introduction

Scheduling an event for a group of people is a notoriously frustrating task; it tends to be tedious, time consuming, and often leads to suboptimal schedules. One reason the problem has resisted a solution is that it has many dimensions, some of them easy to put one's finger on such as the length of the scheduling process, but others more subtle such as game-theoretic manipulability of the procedure. It is not surprising therefore that different work in the area has focused on different dimensions of the problem (we discuss this work in Section 2). While our work as a whole attempts to tackle the problem in its entirety, in this paper we too focus on specific dimensions.

We anchor the discussion in Doodle[1], which is today the most commonly used internet tool for group scheduling. Doodle falls in the class we call *Single-Proposer Mechanisms* (SPMs), in which only one person (the convener) sends out proposals, and invitees respond. Specifically, in Doodle the convener sends a list of possible time slots to the invitees, they respond individually by checking off the slots that are acceptable to them, and based on the responses the convener selects a time slot and announces it to the invitees. Importantly, invitees can see the responses by those who have already responded.

Doodle has several advantages, one of which is simplicity, but it also has many drawbacks. Some of these we will ignore in this work, including vulnerability to procrastination, manipulability, and privacy violation. In this paper we will assume that all invitees are prompt, honest and trusting, since even in this idealized setting the remaining dimensions of the problem – which in a sense are the more basic – pose interesting challenges, and yet are amenable to algorithmic solutions. We will focus on two important dimensions:

Time (T): Length of the scheduling process.

Effort (E): The effort required by the invitees in the process.

The T and E dimensions together will provide an aggregate cost dimension of the scheduling process. Even before we set it up as a formal problem, clearly Doodle lies on one extreme of the two-dimensional T-E Pareto frontier: the process is very short (under the assumption of promptness), but invitees must potentially examine a large set of time slots.[2] On the other extreme

[1] `http://www.doodle.com`

[2] We note in passing that one could hope to eliminate the tedium by granting the scheduling program access to the calendars, and indeed a few such systems exist. But this tends to not be a viable approach in practice, for several reasons: Calendars often under-represent peoples actual commitments, some people have privacy concerns, and more simply you cannot assume that all invitees will be part of this hypothetical system.

would be the "One-at-a-time" mechanism, in which the convener tests individual slots sequentially until one is acceptable by all invitees. This suggests a generalization of Doodle to the class of mechanisms we will call B-Doodle (for Batched Doodle), in which the proposer sends out different batches of slots until in one of the batches an acceptable slot is found. Every B-Doodle mechanism is defined by its batching scheme; clearly Doodle itself is a special case of B-Doodle with all time slots in a single batch. When there is a known tradeoff between T and E, they define an overall cost function, C (for example via their linear combination, C = αT + E). When such a cost function exists (and in most of this paper we will assume it does), and the underlying prior probability of availability of invitees is known, we propose an efficient algorithm for computing the optimal batching scheme; we call the resulting mechanism B-Doodle*. We will show in simulations that in many realistic situations B-Doodle* outperforms Doodle, and often substantially so.

The rest of the paper is organized as follows. In Section 2 we discuss previous related work. In Section 3 we explore the class of B-Doodle mechanisms and underlying assumptions in our model. In Section 4 we provide an efficient algorithm for finding optimal B-Doodle mechanism. In Section 5 we show experimental results of B-Doodle mechanisms and point out inefficiency of Doodle. We conclude in Section 6 with a summary of the results, what we can learn from them, and possible directions for future work.

2 Related Work

Given the importance and multi-dimensionality of group scheduling it is not surprising that much research has been devoted to it, including in AI. Early work in AI (cf., [Jennings *et al.*, 1995; Sen and Durfee, 1998]) had a heuristic, systems-oriented flavor to it, with more emphasis on the system description than on analysis. Some work (cf., [Ephrati *et al.*, 1994]) tackled incentive issues, adopting a game theoretic approach. Another strand in AI has focused on applying machine learning to group scheduling. (cf., [Mitchell *et al.*, 1994; Maes, 1994; Crawford and Veloso, 2005]. In addition to these strands of domain-independent group scheduling, there has been work on group scheduling in specific domains (cf., [Cowling *et al.*, 2001; Chia *et al.*, 1998; Neiman *et al.*, 1994; Decker and Li, 1998; Hannebauer and Müller, 2001].

All of the above directions are relevant to the general problem of group scheduling, but not directly to this paper, and given the space constraints we will not discuss them further. Instead let us discuss in more detail the most closely related work of which we are aware, which is [Garrido and Sycara, 1996] and [Franzin *et al.*, 2004]. These papers are superficially different, as their main focus is the impact of privacy considerations (specifically, different levels of calendar sharing in [Garrido and Sycara, 1996] and different levels of privacy loss due to the information revealed during the scheduling process in [Franzin *et al.*, 2004]) on time and quality, whereas we concentrate on the interaction between time and effort. But this is not the main differentiator, since their notion of privacy level is tied to the number of slots responded to before a schedule is found, which is close to our effort measure. A bigger difference lies in the mechanisms considered; we explore the space of *Single-proposer Mechanisms*, where as they explore two specific mechanisms that fall outside the scope of SPMs. And perhaps the biggest difference lies in the nature of the results; [Garrido and Sycara, 1996; Franzin *et al.*, 2004] do not propose optimal mechanisms, and do not compare their mechanisms to common web-based solutions such as Doodle (nor could they have, since no such solutions existed at the time).

3 Models and Assumptions

In this section, we focus on the *Time-Effort* dimensions. *Time* captures the duration of the scheduling process and *Effort* measures how much effort is required by each invitee during the process. We start this section by defining the group scheduling problem and the class of B-Doodle mechanisms. We also define *Time*, *Effort*, and *Cost* in the context of B-Doodle mechanisms.

3.1 The Class of B-Doodle Mechanisms

Consider the following group scheduling problem. There are I invitees and S time slots. An invitee knows her availability for all S time slots and it is her private information. The sole convener tries to find a feasible time slot by asking invitees for their availability; the convener has a prior definition of feasibility of time slot, which we formally define later.

In the class of B-Doodle mechanisms, the convener iteratively sends out batches of time slots (possibly just one batch), and invitees respond with their respective availability. The scheduling process terminates if a feasible time slot is found, or after all S slots have been proposed. Given a B-Doodle mechanism, we define *Time* of the mechanism to be the number of iterations and *Effort* be the total number of time slots proposed during the process. In this work, we define *Cost* to be their linear combination (α*Time* + *Effort*) where $\alpha > 0$.

3.2 Probabilistic Assumptions

To analyze the efficiency of B-Doodle mechanisms, we make probabilistic assumptions about availability of invitees. For each invitee and each time slot, the invitee is available with probability p and unavailable with $(1-p)$, identically and independently distributed for all invitees and all time slots. p is called the probability of availability.

Definition 1 (r-infeasibility)**.** Given a constant r where $0 \leq r \leq 1$, we say that a time slot is *r-feasible* if at least $\lceil r \cdot I \rceil$ invitees are available at the time slot. For a given time slot, let $f(r)$ be the probability that the time slot is not *r-feasible*. We call $f(r)$ the probability of *r-infeasibility*.

We assume $r = 1$ in most parts of this work (that is, all invitees are required to attend), and simply write f and *feasible* instead of $f(r)$ and *r-feasible*. Later we will discuss how these definitions can be generalized.

Given the underlying probabilistic assumptions, we can compute the expected *Cost*, *Time*, and *Effort* of B-Doodle mechanisms. Let us first present an example of the Pareto frontier on the T-E dimensions where $(I = 4, S = 6, p = .8)$. There are 32 ways to divide 6 time slots into batches. Figure 1 shows all 32 B-Doodle mechanisms.

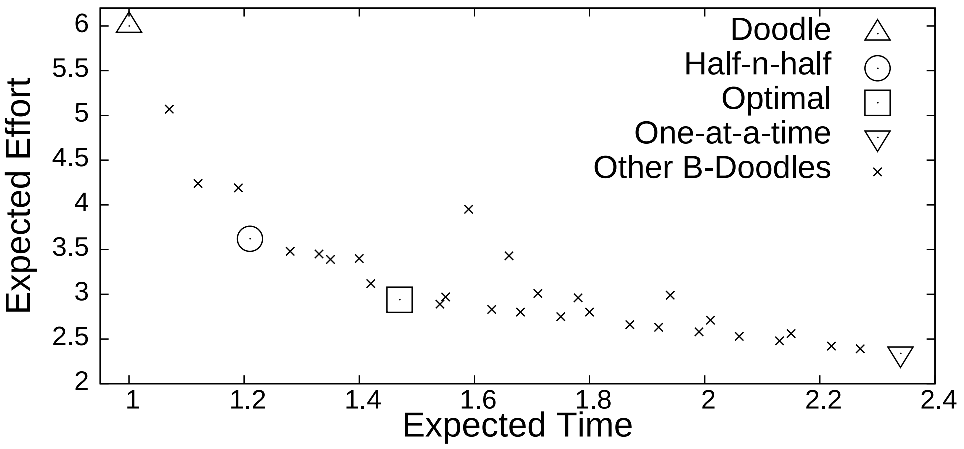

Figure 1: Pareto Frontier of the Time-Effort dimensions.

Far to the left, we see 'Doodle' with the smallest expected *Time*, but with the largest expected *Effort*. Far to the right, we see the 'One-at-a-time' mechanism with opposite properties. The 'Half-n-half' mechanism uses two batches of equal size and is placed somewhere in-between. Given a specific cost function, we can find the respective optimal B-Doodle mechanism in this Pareto Frontier. The 'Optimal' mechanism (that uses three batches of equal size) is optimal when $\alpha = 2$ (thus *Cost* = 2*Time* + *Effort*). As α changes, the optimal mechanism may also change. For instance the 'Half-n-half' mechanism is optimal when $\alpha = 3$, while Doodle is optimal when $\alpha \geq 14.5$.

4 Optimal Mechanism: B-Doodle*

Consider a batching scheme of some B-Doodle mechanism, $\langle b_1, b_2, \ldots, b_m \rangle$ where b_j denotes the number of time slots in j-th batch. The expected cost can be computed if we know the probability of terminating after each iteration – let $P(j)$ be the probability of terminating after j-th batch. Also let s_j denote the total number of time slots in first j batches with $s_0 = 0$. The expected cost C_α of the batching scheme is given by:

$$C_\alpha(\langle b_1, \ldots, b_m \rangle) = \sum_{j=1}^{m} P(j) \cdot (\alpha \cdot j + s_j) \tag{1}$$

Given (I, S), the optimization problem is to find the batching scheme $\langle b_1^*, \ldots, b_{m^*}^* \rangle$ such that C_α in Equation 1 is minimized. A simple, naive way is to try all possible batching schemes of S time slots, but this is inefficient as there are 2^{S-1} such batching schemes. The following theorem leads us to an efficient algorithm that finds the optimal batches.

Theorem 1. *Let $C_\alpha^*(S)$ denote the optimal expected cost given T slots where $C_\alpha^*(0) = 0$. Both C_α and C_α^* have recurrence relations.*

$$C_\alpha(\langle b_1, \ldots, b_m \rangle) = (1 - P(1)) \cdot C_\alpha(\langle b_2, \ldots, b_m \rangle) + (\alpha + b_1)$$

$$C^*_\alpha(S) = \min_{1 \le b_1 \le S} (1 - P(1)) \cdot C^*_\alpha(S - b_1) + (\alpha + b_1) \tag{2}$$

In both recurrence relations, the term $(\alpha + b_1)$ captures the cost of sending out the first batch b_1; whether a *feasible* schedule is found or not in the first batch, this cost must occur. If the first batch fails with probability $(1 - P(1))$, then does occur the extra cost of sending out the remaining batches $\langle b_2, \ldots, b_m \rangle$ of size $(S - b_1)$.

We now describe the algorithm that efficiently finds the optimal batches. Recall that $f(r)$ (or simply f) is the probability that a given time slot is not *r-feasible*. The algorithm first computes f and powers f^b for all $b \le S$ as a pre-processing step in time $O(S + \log I)$. Next, the algorithm uses dynamic programming to compute $C^*_\alpha(S)$ in $O(S^2)$ time, using Equation 2 and $P(1) = (1 - f^{b_1})$. While computing C^*_α, the algorithm also stores the respective optimal b_1 values, and traces back the optimal batches $\langle b^*_1, b^*_2, \ldots, b^*_{m^*} \rangle$. B-Doodle* simulates this algorithm to find the optimal batches and sends them out accordingly.

Remark. We can generalize $(r = 1)$ to $(r \le 1)$ and introduce the probability of availability per invitee, $\{p_i\}$. For general $(r \le 1)$ and $\{p_i\}$, dynamic programming can be used to compute $f(r)$ in $O(I^2)$ time (as opposed to $O(\log I)$ when $r = 1$ and common p for all invitees). With this, the overall time complexity of the algorithm in B-Doodle* changes from $O(S^2 + \log I)$ to $O(S^2 + I^2)$, but all equations and theorems in this section still hold. We omit proofs.

5 Experimental Results

In this section, we compare Doodle to B-Doodle* in realistic settings. For an experimental choice, we used $\alpha = 2$ while we varied (I, S, p, r).

If there is only one time slot, then Doodle is trivially optimal. On the other hand, if S is large, Doodle is suboptimal because it requires invitees to examine too many time slots. Hence, it is interesting to know for what ranges of S, Doodle is suboptimal. Given (I, p), let $C^D_\alpha(S)$ be the cost of Doodle and $C^*_\alpha(S)$ be the expected cost of B-Doodle*. We want to find the critical point $S^*(I, p) = \arg\min_{S \ge 1}[C^D_\alpha(S) < C^*_\alpha(S)]$; note that Doodle is suboptimal for all $S \ge S^*(I, p)$ given (I, p).

In Table 1, we show $S^*(I, p)$ for various (I, p). Each entry shows the value of $S^*(I, p)$ for specific (I, p). For instance, $S^*(2, .8) = 3$, which implies that Doodle is suboptimal for all $S \ge 3$ given that $I = 2$ and $p = .8$. Hence, the smaller $S^*(I, p)$ is, the less practical Doodle is for the corresponding (I, p). In particular we find 8 entries (highlighted in boldface) in which $S^*(I, p) \le 15$. For the corresponding (I, p) values, Doodle is suboptimal if 15 time slots are proposed. This result shows that Doodle is suboptimal for a small number of invitees with high or moderate availability (under our probabilistic assumptions and experimental choices).

$S^*(I,p)$	$I = 2$	$I = 4$	$I = 6$	$I = 10$	$I = 15$
$p = .8$	**3**	**4**	**5**	**8**	**15**
$p = .5$	**5**	**11**	22	90	> 300
$p = .2$	**14**	70	> 300	> 300	> 300

Table 1: Critical point $S^*(I, p)$ when $r = 1$. Entries are highlighted in boldface when Doodle is suboptimal given $S = 15$.

We do not only want to know when Doodle is suboptimal, but also want to know how inefficient Doodle is in realistic situations. Given (I, p, S), we define the efficiency of Doodle, e_D, as the ratio of the optimal expected cost to the cost of Doodle: $e_D = C^*_\alpha(S)/C^D_\alpha(S)$. Table 2 shows e_D values for the same settings of (I, p) as in Table 1 (we used $S = 15$ time slots in this experiment). For instance in the first row, $e_D = .270$ when $I = 2$, which implies that Doodle is very inefficient when $I = 2$ and $p = .8$. Notice how the values of e_D vary across the table; the smaller I is and the higher p is, the lower e_D is and the more inefficient Doodle is. In particular, we find four entries (highlighted in boldface), for which the efficiency of Doodle is below .750. Interestingly, in the third row with $p = .2$ Doodle seems very efficient for all I; in this case it is difficult to find a feasible time slot, and multiple batches may incur an extra cost.

e_D	$I = 2$	$I = 4$	$I = 6$	$I = 10$	$I = 15$
$p = .8$	**.270**	**.361**	**.486**	.777	.986
$p = .5$	**.502**	.904	1	1	1
$p = .2$	.970	1	1	1	1

Table 2: Efficiency of Doodle when $S = 15$ and $r = 1$. Entries are highlighted in boldface when the efficiency of Doodle is below .750.

If we relax the condition and require $(r = .7)$ instead of $(r = 1)$, Doodle becomes even more inferior to B-Doodle*. Table 3 shows e_D values for the same (I, p) values as in Table 2; the only difference between the two tables is the choice of r. Note that the first column of the two tables are identical because $r = .7$ requires all invitees to be available when $I = 2$. As we compare the entries of the two tables, we observe that in Table 3, the suboptimality of Doodle is more pronounced for a larger domain of (I, p). In Table 2 the efficiency of Doodle is below .750 in four entries, while in Table 3 the efficiency of Doodle is below .750 in eight entries – twice as many. Also notice that e_D values are surprisingly low in the first row of Table 3 across all columns. This shows that regardless of the number of invitees, Doodle is significantly inefficient when the invitees are highly available $(p \geq .8)$ and the attendance requirement is relaxed $(r = .7)$.

e_D	$I = 2$	$I = 4$	$I = 6$	$I = 10$	$I = 15$
$p = .8$	**.270**	**.215**	**.267**	**.201**	**.211**
$p = .5$	**.502**	**.434**	.772	**.628**	.913
$p = .2$	.970	1	1	1	1

Table 3: Efficiency of Doodle when $S = 15$ and $r = .7$. Entries are highlighted in boldface when the efficiency of Doodle is below .750.

In all our experiments, we used $\alpha = 2$, but we ran the same set of experiments with larger α as well. Given large $\alpha \geq 20$ we observed that Doodle is optimal when $(p \leq .2)$ regardless of r. However, we also observed that Doodle is still inefficient when $(p \geq .8)$ and $(r < 1)$ for such large α.

6 Discussion and Future Work

The main contribution of this work is the proposal of an algorithmic approach to the group scheduling problem. We began by identifying and formally defining two important dimensions of optimality in scheduling problem: *Time*and *Effort*.

We presented an example of the Pareto Frontier of B-Doodle mechanisms on the *Time-Effort* dimensions. We also described an efficient algorithm for finding the optimal batches on the *Cost* dimension. Using the algorithm, we provided B-Doodle*, the optimal mechanism from the *Time-Effort* perspective. In simulations, we showed that Doodle is substantially inefficient in many realistic settings, particularly when attendance requirement is relaxed (under our assumptions and experimental choices). We also reported that even if we change our choice of α to a larger value, Doodle is still suboptimal to a great extent for a small number of invitees with high availability and with lower attendance requirement. Note that we defined *Cost* as a linear combination of *Time* and *Effort*, but other definitions of *Cost* can be studied for future work.

Another direction for future work is towards analyzing the class of *Multi Proposer Mechanisms* (MPMs) on the same dimensions of optimality. Intuitively, a time slot proposed by an invitee is more likely to be feasible and than an arbitrary time slot, not to mention that it is highly preferred by at least one invitee. Hence, Multi-proposer Mechanisms can be very efficient in realistic situations.

Finally, we assumed honesty, promptness, and trust of invitees, but these are strong assumptions. In particular, promptness assumes that invitees do not procrastinate and do not deliberately delay responses. We argue that if the scheduling process is efficient on the cost dimensions, it will reduce the procrastination of invitees because it is quick but not costly. In practice, an invitee may have an incentive to delay her response or to lie about her availability or preferences. Practical mechanisms should not be vulnerable to such strategic behavior of invitees, and this is yet another direction for future work.

References

[Chia *et al.*, 1998] M.H. Chia, D.E. Neiman, and V.R. Lesser. Coordinating asynchronous agent activities in a distributed scheduling system. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98)*, 1998.

[Cowling *et al.*, 2001] Peter Cowling, Graham Kendall, and Eric Soubeiga. A hyperheuristic approach to scheduling a sales summit. In Edmund Burke and Wilhelm Erben, editors, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pages 176–190. Springer Berlin Heidelberg, 2001.

[Crawford and Veloso, 2005] E. Crawford and M. Veloso. Learning to select negotiation strategies in multi-agent meeting scheduling. *Progress in Artificial Intelligence*, pages 584–595, 2005.

[Decker and Li, 1998] K. Decker and J. Li. Coordinated hospital patient scheduling. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, pages 104–111. IEEE, 1998.

[Ephrati *et al.*, 1994] E. Ephrati, G. Zlotkin, and J.S. Rosenschein. A non-manipulable meeting scheduling system. In *Proceedings of the 13th international workshop on distributed artificial intelligence*, pages 105–125, 1994.

[Franzin *et al.*, 2004] M. S. Franzin, F. Rossi, E. C. Freuder, and R. Wallace. Multi-agent constraint systems with preferences: Efficiency, solution quality, and privacy loss. *Computational Intelligence*, 20(2):264–286, 2004.

[Garrido and Sycara, 1996] L. Garrido and K. Sycara. Multi-agent meeting scheduling: Preliminary experimental results. In *Proceedings of the Second International Conference on Multiagent Systems*, pages 95–102, 1996.

[Hannebauer and Müller, 2001] M. Hannebauer and S. Müller. Distributed constraint optimization for medical appointment scheduling. In *Proceedings of the fifth international conference on Autonomous agents*, pages 139–140. ACM, 2001.

[Jennings *et al.*, 1995] N. R. Jennings, A. J. Jackson, and London E Ns. Agent-based meeting scheduling: A design and implementation, 1995.

[Maes, 1994] P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40, 1994.

[Mitchell *et al.*, 1994] T.M. Mitchell, R. Caruana, D. Freitag, J. McDermott, D. Zabowski, et al. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):80–91, 1994.

[Neiman *et al.*, 1994] D.E. Neiman, D.W. Hildum, V.R. Lesser, T. Sandholm, et al. Exploiting meta-level information in a distributed scheduling system. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 394–394. JOHN WILEY & SONS LTD, 1994.

[Sen and Durfee, 1998] S. Sen and E.H. Durfee. A formal study of distributed meeting scheduling. *Group Decision and Negotiation*, 7(3):265–289, 1998.